## Intro & linear models (INLP ch. 1 & 2)

### CS 685, Spring 2021

Advanced Topics in Natural Language Processing <http://brenocon.com/cs685> [https://people.cs.umass.edu/~brenocon/cs685\\_s21/](https://people.cs.umass.edu/~brenocon/cs685_s21/)

### Brendan O'Connor

College of Information and Computer Sciences University of Massachusetts Amherst

*[Material from Eisenstein (2019)]*

## Levels of linguistic structure



## **NLP as linguistic prediction**



- x: Text, y: Sentiment label
	- x: Text, y: Syntax tree A. ICAL, y. Oynical lice
- x: English text, y: Chinese text translation
- x: Book, y: Major characters & their relationships not more linguistics.  $T$ . Boon, y. major onaraotore a thoir rolations ipo

# **NLP modeling**



- Learning: find a good θ from data (we we need learning at all?)
- **Modeling:** design important ling. phenomena into Ψ the 2017 conference on Neural Information Processing Systems. He was advocating for more learning theory, not more linguistics.
- Reuse search/learning optimization methods for for many different NLP problems numbered by parentheses.

# **NLP modeling**

- **Pred./Search:** Scorer (model) Params  $\hat{\bm{y}}$  $\bm{y} = \argmax \Psi(\bm{x}, \bm{y}; \bm{\theta}) \text{,}$  $y \in \mathcal{Y}(x)$ Params Input (text) **Predicted** output **Output** Possible "I<sup>I</sup>P<sup>UL</sup> candidate" outputs
	- $\tau_{\text{e}}$  is the input of a set  $\tau_{\text{e}}$  of a set  $\tau_{\text{e}}$  of a set  $\tau_{\text{e}}$  of  $\tau_{\text{e}}$   $\tau_{\text{e}}$  of  $\tau_{\text{e}}$  • Today: Linear models Ψ, BOW **x** & **f**, multiclass **y**
	- *Models: Naive Bayes and Logistic Reg* • Models: Naive Bayes and Logistic Regression
		- Learning: (regularized) MLE
- $\bullet$   $M$ adnasdav: Naural natwork  $\Psi$ • Wednesday: Neural network Ψ
	- **Throughout the number of the number of** • Later: sequential **x**
- Later: structured output y

## Linear classification models

- Assume classification problem:
- Input text **x**  $\blacksquare$  Imput text  $\boldsymbol{x}$
- Output discrete yeY,  $|Y|=K$
- Scoring function is a dot product of weight vector **θ** and a vector-valued feature function **f** 
	- **x** is a representation of the text. What to use?
- **f** computes features to score the candidate output. What features to use?

$$
\text{Input} \quad \text{Output} \\ \text{(text)} \\ \Psi(x, y) = \theta \cdot f(x, y) = \sum_{j} \theta_j f_j(x, y)
$$

## **Bag of words representation** 14 *CHAPTER 2. LINEAR TEXT CLASSIFICATION*



- **x** is a vector, representing counts of words in the text
- Vocabulary **V**: set of all word types under consideration  $\begin{bmatrix} 0 & \cdots & \cdots & 0 \end{bmatrix}$  is the compatibility with the label FICT of the label FICTION, for the label FICTION,  $\begin{bmatrix} 0 & \cdots & \cdots & 0 \end{bmatrix}$ we might a positive score to the whole to the whole score to the whole whole to the word a negative score to the word  $\frac{1}{2}$  of whateas to the word  $\frac{1}{2}$  of whateas to the word  $\frac{1}{2}$  of whateas to the word  $\frac{1$ 
	- *x*
	- Len $(x) = V$
- **b**  $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$  worst **information!!** Yet is useful…  $\begin{array}{c|c|c|c|c} \text{worst of} & \text{\color{red}o} & \text{\color{red}o} & \text{\color{red}o} & \text{BOW ignores order} \end{array}$
- $\overline{y}$  ,  $\overline{y}$  ,  $\overline{y}$ of  $\frac{1}{2}$ ,  $\frac{1}{2}$  yxt **contracts** and the contract of th **Exercise** *i*. Weights for each possible and  $\alpha$  and  $\alpha$  and  $\alpha$  and  $\alpha$  and  $\alpha$  are  $\alpha$   $\alpha$   $\beta$

$$
\Psi(\boldsymbol{x},y) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x},y) = \sum_j \theta_j f_j(\boldsymbol{x},y)
$$

### As the notation suggests, *f* is a function of two arguments, the word counts *x* and the 14 *CHAPTER 2. LINEAR TEXT CLASSIFICATION* Bag of words: linear model

• One feature for each word and class pair. of the teature vector  $\bullet$ 

$$
f_j(\boldsymbol{x}, y) = \begin{cases} x_{\text{whale}}, & \text{if } y = \text{FICTION} \\ 0, & \text{otherwise} \end{cases}
$$

- And another for y=NEWS, y=GOSSIP, y=SPORTS in the set of possible labels. The set of possible labels weight  $\tau$  then scores then scores the compatibility of  $\tau$ (and all other output classes)<br><del>T</del>i is to predict a label *y*
	- Thus we compute a score (*x*<sup>*y*</sup>), which is a scalar measure of the compatibility between the co
- the word **number of the label FICTION.** • **f**:  $X \times Y \rightarrow R$ <sup>VK</sup>
	- $\bullet$   $\theta \in \mathbf{R}$ <sup>VK</sup>  $\theta$   $\theta$   $\in$  R<sup>VK</sup>

$$
\Psi(\boldsymbol{x},y) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x},y) = \sum_j \theta_j f_j(\boldsymbol{x},y)
$$

## Bag of words: linear model 14 *CHAPTER 2. LINEAR TEXT CLASSIFICATION*



$$
\Psi(\boldsymbol{x},y) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x},y) = \sum_j \theta_j f_j(\boldsymbol{x},y)
$$

## How to set parameters?

- Could use deterministic 1 and 0 weights implicit in lexicon/dictionary/keyword methods (e.g. racial slur blacklist, sentiment lexicons, etc.) **P** Could use deterministic 1 and 0 weights —
	- But if you have labeled data, typically **supervised learning** is better.
- Labeled data: "gold-standard" (text, label) pairs

$$
\{(\boldsymbol{x}^{(i)},y^{(i)})\}_{i=1}^N
$$

- **dentically distributed (iID)** (*IIND linear, probabilistic models today <sup>i</sup>*=1 p*X,Y* (*x*(*i*) *, y*(*i*) • Two linear, probabilistic models today
	- **1999 International to Angles**<br>• Naive Bayes • Naive Bayes
	- Logistic Regression

### Naive Bayes 2.2 Name Bayes Bayes<br>2.2 Name Bayes B<br>2.2 Name Bayes B  $2.2$  N/z

• Assume a model of both text and labels<br>(each doc i i d ) (each doc i.i.d.) **dentically distributed (Benedicted (II)** (See *§* A.3). The sees *§* A.3). The sees  $\alpha$  of the entire The **joint probability** of a bag of words *x* and its true label *y* is written p(*x, y*). Suppose *<u>id</u>* identity **dentically distributed (IP)** (see *§ 3,5)*. The joint probability of the entire e

$$
\mathbf{p}(\boldsymbol{x}^{(1:N)}, y^{(1:N)}) = \prod_{i=1}^{N} \mathbf{p}_{X,Y}(\boldsymbol{x}^{(i)}, y^{(i)})
$$

- $\bullet$  or one of a supersimate that does the set of the dominate in the label and p<sub>(x,y)</sub> is a **generative model**. Thas a story of now both the fabel and document text was generated • p(x,y) is a **generative model**: has a story of how both the label and  $\mathcal{N}$  does this have to do with classification? One approach to classification is to set  $\mathcal{N}$
- generating text is a.k.a. **language model** other LMs are **model** other LMs are **may** used a lot in NLP
	- Once we have a model, we can do:<br>1 Learning: fit p(*x*, *i*)'e parameters to
		- $\mathsf{S} \restriction$ ˆ arameters to training data (using MLE) 1. Learning: fit p(x,y)'s parameters to training data (using MLE) ve nave a mouer,<br>ning: fit n(v v)'e nr ,<br>,<br>, *, y*(1:*N*)  $\mathsf{C}$   $\mathsf{C}$   $\mathsf{C}$   $\mathsf{C}$   $\mathsf{C}$
		- 2. Prediction ("Search"): infer labels on new documents (using Bayes Rule) *N*

### NB generative model 18 *CHAPTER 2. LINEAR TEXT CLASSIFICATION*

Algorithm 1 Generative process for the Naïve Bayes classification model

 $D_1$  *CHAPTER 2. CALCARDER 2. CALCARDER 2. LINEAR TEXT CLASSIFICATION* **for** Instance  $i \in \{1, 2, \ldots, N\}$  **do:** Draw the label  $y^{(i)} \sim$  Categorical( $\mu$ ); Draw the word counts  $x^{(i)} \mid y^{(i)} \sim \text{Multinomial}(\phi_{y^{(i)}}).$ 

**Algorithm 2** Alternative generative process for the Naïve Bayes classification model

**for** Instance  $i \in \{1, 2, \ldots, N\}$  do: Draw the label  $y^{(i)} \sim$  Categorical( $\mu$ ); **for** Token  $m \in \{1, 2, ..., M_i\}$  **do**: Draw the token  $w_m^{(i)} \mid y^{(i)} \sim$  Categorical $(\boldsymbol{\phi}_{y^{(i)}})$ . also assume that the documents don't affect each other: if the word *whale* appears

- Types vs. Tokens in document *i* = 7, that does not make it any more or less likely that it will appear
- Generative probability notation: better to think of them as sequences of tokens, *w*. If the tokens are generated from a for any of probability *µ*. Categorie
	- a ~ Distrib(theta): "Random variable a is sampled according to distribution Distrib, parameterized by theta." models that are identical, except for a scaling factor that does not depend on the label or a ~ Distrib(theta): "Random variable a is sampled accord **2.2.2 Prediction** each label, so that the probability of drawing at the probability of  $\mathcal{Y}$ *V* alstribution Distrib, parameterized by theta.<br>,
- Parameters **and** *Parameters <sup>y</sup>*2*<sup>Y</sup> <sup>µ</sup><sup>y</sup>* = 1 and *<sup>µ</sup><sup>y</sup>* <sup>0</sup>*,* <sup>8</sup>*<sup>y</sup>* <sup>2</sup> *<sup>Y</sup>*: each label's probability is non-negative, and the
	- **µ**k: prior probability of class k  $\mu$ <sub>k</sub>: prior probability of class k
	- $\bullet$   $\Phi_{k,w}$ : probability word w gets generated under doc class  $k$  $\bm{y}$ rd w gets generated under doc class *<sup>x</sup>*(*i*) *<sup>|</sup> <sup>y</sup>*(*i*) , this line indicates that the word counts are conditioned on the label, so  $\mathsf{p}_{\mathsf{K},\mathsf{w}}$ : probability word w gets generate
- "Naive": each word token is generated independently.

#### MB prediction<br>
MB prediction the parameters. models that are identical, except for a scaling factor that does not depend on the label or the parameters.

**•** First assume we have parameters. How do we predict the label given text? Chose the one with *highest posterior probability*  $p(y | x) = p(x, y) / p(x)$ **2.2.2 Prediction**  $T$ *y*ˆ = argmax log p(*x, y*; *µ,* ) [2.14]

$$
\hat{y} = \operatorname*{argmax}_{y} \log p(\boldsymbol{x}, y; \boldsymbol{\mu}, \boldsymbol{\phi})
$$

$$
= \operatorname*{argmax}_{y} \log p(\boldsymbol{x} | y; \boldsymbol{\phi}) + \log p(y; \boldsymbol{\mu})
$$

$$
\log p(\boldsymbol{x} \mid y; \boldsymbol{\phi}) + \log p(y; \boldsymbol{\mu}) = \log \left[ B(\boldsymbol{x}) \prod_{j=1}^{V} \boldsymbol{\phi}_{y,j}^{x_j} \right] + \log \mu_y
$$

$$
= \log B(\boldsymbol{x}) + \sum_{j=1}^{V} x_j \log \phi_{y,j} + \log \mu_y
$$

- ear model (see text).<br><del>fwerde end eless pri</del> *V* • This can be shown to be a linear model (see text).
	- *x<sup>j</sup>* log *y,j* + log *µ<sup>y</sup>* [2.17] Parameters = log probs of words and class priors
	- *j*=1 = log *B*(*x*) + ✓ *· f*(*x, y*)*,* [2.18] • Features = count of word under candidate class; candidate class

### NB learning **2.2.3 Estimation**

**•** Intuitively, relative frequency estimation sounds good:

$$
\phi_{y,j} = \frac{\text{count}(y,j)}{\sum_{j'=1}^{V} \text{count}(y,j')} = \frac{\sum_{i:y^{(i)}=y} x_j^{(i)}}{\sum_{j'=1}^{V} \sum_{i:y^{(i)}=y} x_{j'}^{(i)}}
$$

 $\bullet$  This has a deeper theoration hasisl **maximum likelihood estimate**: the estimate that maximizes the probability p(*x*(1:*N*) • This has a deeper theoretical basis!

#### NB learning: MLE we have a dataset of *<sup>N</sup>* labeled instances, *{*(*x*(*i*) **dentically distributed (IID)** (See *§* A.3). The *§* A.3). The see *§* A.3). The set  $\mathbf{A}$ dataset, written p(*x*(1:*N*) *, y*(1:*N*) ), is equal to Q*<sup>N</sup> <sup>i</sup>*=1 p*X,Y* (*x*(*i*)

• **Maximum Likelihood Estimation**: choose params that give highest likelihood to the training data  $M$ ovimum Likolihaad Estimation: ohooso paramo **THEATHTENT ENTERNATE DOCUTATION**<br>that aive biobaat likelihood to the training dota ments. This is known as **maximum likelihood estimation**:

$$
\hat{\theta} = \operatorname*{argmax}_{\theta} p(\boldsymbol{x}^{(1:N)}, y^{(1:N)}; \theta)
$$
\n
$$
= \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}, y^{(i)}; \theta)
$$
\n
$$
= \operatorname*{argmax}_{\theta} \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, y^{(i)}; \theta)
$$
\n
$$
\mathcal{L}(\theta)
$$
\n
$$
\log \text{likelihood function}
$$

## NB learning: MLE Equation 2.21 defines the **relative frequency estimate** for . It can be justified as a

• Choose  $\phi, \mu$  to maximize log-likelihood

$$
\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \log p_{\text{mult}}(\boldsymbol{x}^{(i)}; \boldsymbol{\phi}_{y^{(i)}}) + \log p_{\text{cat}}(y^{(i)}; \boldsymbol{\mu})
$$

• Under the sum-to-1 constraints

$$
\sum_{j}^{V} \phi_{y,j} = 1 \quad \forall y \qquad \qquad \sum_{k}^{K} \mu_k = 1
$$

*i*=1 *i*=1 ===> intuitive relative frequency estimates! • Calculus with Lagrange multipliers

#### Bias-variance tradeoffs appeared in a work of fiction. But choosing a value of FICTION*,molybdenum* = 0 would allow this single feature to completely veto a label, since p(FICTION *| x*)=0 if *xmolybdenum >* 0.

- Does MLE overfit or underfit?  $\rho$  be peak that  $\rho$  in the training set, we concern the set of  $\rho$
- Laplace smoothing: add a pseudocount, lution is to **smooth** the probabilities, by adding a "pseudocount" of ↵ to each count, and

$$
\phi_{y,j} = \frac{\alpha + \text{count}(y, j)}{V\alpha + \sum_{j'=1}^{V} \text{count}(y, j')}
$$

- (Why Vα ?) <sup>11</sup> The pseudocount ↵ is a **hyperparameter**, because it
- $\alpha$  -varme  $--$  ? •  $\alpha = >$ large  $==$ >?

## What about class priors? Equation 2.21 defines the **relative frequency estimate** for . It can be justified as a

• Choose  $\phi, \mu$  to maximize log-likelihood

$$
\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\mu}) = \sum_{i=1}^{N} \log p_{\text{mult}}(\boldsymbol{x}^{(i)}; \boldsymbol{\phi}_{y^{(i)}}) + \log p_{\text{cat}}(y^{(i)}; \boldsymbol{\mu})
$$

- **µ** is set to class frequencies in training data. Is this realistic? on the parameters . Since p(*y*) is constant with respect to , we can drop it: *L*() =<sup>X</sup>  $\bigcap_{x \in \mathbb{R}} \bigcap_{x \in \mathbb{R$  $\overline{\phantom{a}}$ 
	- [See our paper! Keith and O'Connor, 2018] log pmult(*x*(*i*) ; *y*(*i*) ) = <sup>X</sup> *i*=1 *j*=1 *<sup>j</sup>* log *y*(*i*)*,j ,* [2.23]

## Cond. indep. is a problem 14 *CHAPTER 2. LINEAR TEXT CLASSIFICATION*

- We can do better than BOW features by feature engineering lots of little variants of words and phrases with the compatibility with the label of the label with the label of the l
- e.g. ngram features ... character n-grams ... words with or without lowercasing ... number of digits in the text … number of punctuation marks in the text … etc. Suppose that the lext ... number of punctuation<br>marks in the text , at a marks in the text ... etc.
- Even overlapping features can have useful<br>
× *x*, weights *x*, using the weights *x*, weights *x*, and *x*, weights *x*, we are a *y* 2 *y*, we are a *y* 2 *y*, we are a *y* 2 *y*, we are a *y* 2 *y* 2 *y* 2 *y* 2 *y* 2 predictive value of the compatibility between the companion of the co is to predict a label *y*
- But... does NB do well with repetitive features? features?

$$
\Psi(\boldsymbol{x},y) = \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x},y) = \sum_{j} \theta_j f_j(\boldsymbol{x},y)
$$

## Discriminative learning

- NB (generative) learning chooses params to maximize p(Xtrain, Ytrain), then indirectly gives the linear prediction model.
- But if we just care about prediction, why not directly learn to minimize prediction errors?
	- Perceptron: choose params to minimize **1-0 loss**.
	- SVM: choose it to minimize **hinge loss**
- Logistic regression: choose theta to maximize **conditional log-likelihood** (a.k.a. minimize logistic loss a.k.a. cross-entropy)

### Logistic regression defines the desired **conditional probability** <sup>p</sup>*<sup>Y</sup> <sup>|</sup><sup>X</sup>* directly. Think of ✓ *· <sup>f</sup>*(*x, y*) as a scoring

• Directly define the conditional probability of label given text via the **softmax** of the linear scoring function resulting condition

$$
p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y))}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}, y'))}
$$
 *exp and normalized*

Under contract with MIT Press, shared under CC-BY-NC-ND license. • Learning: choose theta to maximize the conditional log-likelihood,

$$
\log p(\boldsymbol{y}^{(1:N)} \mid \boldsymbol{x}^{(1:N)}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})
$$

$$
= \sum_{i=1}^{N} \boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) - \log \sum_{\boldsymbol{y}' \in \mathcal{Y}} \exp \left(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, \boldsymbol{y}')\right)
$$
Give high scores to observed y(i)

### Logistic loss *i*=1

Negative log-likelihood for one example the sum, we have the (additive inverse of the) **logistic loss**,  $\ell_{\texttt{LOGREG}}(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)}) = -\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y^{(i)}) + \log \sum \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}^{(i)}, y'))$  $y' \in Y$  $\mathcal{L}_{\text{LUGNEC}}(\nu, \omega, \gamma)$  ,  $\mathcal{L}_{\text{LUGN}}$ 



Figure 2.2: Margin, zero-one, and logistic loss functions.

- errors
	- 99% prob for y(i)  $\Rightarrow$

$$
• 1\% prob for y(i)
$$
\n
$$
= > \text{ or } \text{ or } y(i)
$$

 $0\%$ ?

# LogReg learning

- There is no closed form MLE
- But, fortunately, the log-lik is concave (NLL convex)
- Use gradient descent!
	- 1. Calculate gradient equations
	- 2. Use a batch or online gradient algorithm

### Regularization model, *<sup>E</sup><sup>Y</sup> <sup>|</sup>X*[*f*(*x*(*i*) *, y*)], and the observed feature counts *f*(*x*(*i*) vectors are equal for a single instance, there is nothing more to learn from it; when they

- For many NLP feature functions, training data is often linearly separable. Weights diverge to +/- inf often linearly senarable Weights diverge to  $+/-$  inf
	- Regularization is essential. Typically use the L2 norm of weights, resulting in a regularized loss: *, y*)] under the conditional distribution p(*y | x*; ✓). The regulting in a require

$$
L_{\text{LOGREG}} = \!\frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 - \sum_{i=1}^{N}\left(\boldsymbol{\theta}\cdot\boldsymbol{f}(\boldsymbol{x}^{(i)},y^{(i)}) - \log \sum_{y'\in\mathcal{Y}}\exp \boldsymbol{\theta}\cdot\boldsymbol{f}(\boldsymbol{x}^{(i)},y')\right)
$$

## **Summary: NLP prediction**



- Today: Linear models Ψ, BOW **x** & **f**, multiclass **y**  $\tau_{\text{e}}$  is the input of a set  $\tau_{\text{e}}$  of a set  $\tau_{\text{e}}$  of a set  $\tau_{\text{e}}$  of  $\tau_{\text{e}}$   $\tau_{\text{e}}$  of  $\tau_{\text{e}}$
- Models: Naive Bayes and Logistic Regression • *Models: Naive Bayes and Logistic Reg* 
	- Learning: (regularized) MLE
- Wednesday: Neural network Ψ  $\bullet$   $M$ adnasdav: Naural natwork  $\Psi$ 
	- Later: sequential **x Throughout the number of the number of**
- **•** Later: structured output **y**